Raymond Salzmann

EEL3744C

Lab 4

Pre-Lab Questions

1. Which pins on PORTD are used for USARTD0?

   Pins 1,2 & 3 on PORTD are used for USARTD0

2. What is the maximum possible baud you can use for asynchronous communication, if your board runs at 32 MHz? Support your answer.

   The maximum possible baud rate for the board at 32MHz is 2Mbps if the CLK2x = 0 and is 4Mbps if the CLK2x = 1. These values were found from the Fractional Baud Rate section of the 8331 Manual.

3. What is the difference between serial and parallel communication?

   Serial communication can only send one bit of data at a time, Parallel communication can send multiple bits simultaneously.

4. What is the difference between synchronous and asynchronous communication?

   In synchronous communication, the sender and receiver use the same clock signal, while during asynchronous communication the sender provides a synchronization signal to the receiver before the transfer of each message.

5. List the XMEGA's USART registers used in your programs and briefly describe their functions.
   - CTRLB: This register is used to enable the transmitter and the receiver
   - CTRLC: This register controls the timing, parity mode, stop bits and data bits
   - STATUS: This register shows the status of the USART with interrupts and such
   - DATA: This register is the buffer register for both transmitting and receiving
   - BAUDCTRLA: Controls bits [7..0] of the BSEL value
   - BAUDCTRLB: Controls bits [11..8] of the BSEL value & controls the BSCALE value

Problems Encountered

The problems I encountered throughout the lab were few and far between, however I had a large issue with getting the input/output character function to run more than once. I also had some issues with the spacing of characters in Part F.

Future Work/Applications

There are some fun applications to this lab and a lot of practical uses with the creation of a menu and being able to interface with said menu. If I had more time to work on the lab I would try to clean my code up a little more and make some better comments.

Pre-Lab

Part B:

Pseudocode:

1. Set up the Clock for 32MHz
2. Initialize the USART for 460,800 Baud Rate and Transmit Enable
3. Create a subroutine for OUT_CHAR
4. In this subroutine we check if the DATA register is empty so that we can transmit new data
5. Write the character desired to the DATA register
6. Loop Infinitely to keep outputting U

Program Code:

```
; Lab 4 Part B
; Name: Raymond Salzmann
; Section #: 2B04
; TA Name: Keith Fitzgerald
; Description: XMEGA USART TRANSMITTER

.include "ATxmega128A1Udef.inc"

.org 0x0000
	rjmp MAIN

.org 0x200
MAIN:
	call CLOCK_32MHz_SETUP
	call USART_INIT

LOOP:
	ldi r17, 85		;ASCII code for 'U'
	call OUT_CHAR
	rjmp LOOP

;*********************************************************
CLOCK_32MHz_SETUP:
	push r16

SETUP:
	ldi r16, 0x02 ;set bit 1
	sts OSC_CTRL, r16 ;enable OSC_CTRL with the 32 MHz internal Clock

POLL_1:
	lds r16,OSC_STATUS ;load the value of OSC_STATUS to r16
	sbrc r16, 1 ;poll the OSC_STATUS register for stable flag
```

```
        rjmp CONTINUE ;continue once bit 1 is set
        rjmp POLL_1 ;loop until bit 1 is set

CONTINUE:
        ldi r16,0xD8 ;load 'D8' to r16 to enable IOREG
        sts CPU_CCP, r16 ;write IOREG to CPU_CCP

        ldi r16, 0x01 ;load '1' to bit 1
        sts CLK_CTRL,r16 ;store '1' in the CLK_CTRL register to enable the 32 MHz clk

        pop r16
        ret

;************************************************
USART_INIT:
        push r16

        ldi r16, 0x08
        sts PORTD_DIRSET, r16

        ldi r16, 0x06
        sts PORTD_DIRCLR, r16

        ldi r16, 0b00110011  ;Asynchronus, Odd Parity Bit, 1 stop bit and 8 data bits
        sts USARTD0_CTRLC, r16

        ldi r16, 0b10101100                     ;lower byte of the Baud Select value

        sts USARTD0_BAUDCTRLA, r16

        ldi r16, 0b10010001  ;high nibble to 1001 for two's comp of -7, top of BSEL to 0001

        sts USARTD0_BAUDCTRLB, r16

        ldi r16, 0b00001000                     ;Transmit enable
        sts USARTD0_CTRLB, r16

        pop r16

        ret

;************************************************

OUT_CHAR:
        push r16
        push r17

OUT_CHAR_WAIT:
        lds  r16, USARTD0_STATUS
        sbrs r16, 5   ;check if DATA register is empty so that we can transmit next value
        rjmp OUT_CHAR_WAIT                      ;loop to wait for output enable

        sts USARTD0_DATA, r17                   ;store the character in the data register

        pop r17
        pop r16

        ret
```

Part C:

Psuedocode:

1. Use the code from Part B but replace everything in PORT D with PORTC

Program Code:

```
; Lab 4 Part C
; Name: Raymond Salzmann
; Section #: 2B04
; TA Name: Keith Fitzgerald
; Description: MEASURING THE BAUD RATE

.include "ATxmega128A1Udef.inc"

.org 0x0000
	rjmp MAIN

.org 0x200
MAIN:
	call CLOCK_32MHz_SETUP
	call USART_INIT

LOOP:
	ldi r17, 85		;ASCII code for 'U'
	call OUT_CHAR
	rjmp LOOP

;*****************************************************
CLOCK_32MHz_SETUP:
	push r16

SETUP:
	ldi r16, 0x02 ;set bit 1
	sts OSC_CTRL, r16 ;enable OSC_CTRL with the 32 MHz internal Clock

POLL_1:
	lds r16,OSC_STATUS ;load the value of OSC_STATUS to r16
	sbrc r16, 1 ;poll the OSC_STATUS register for stable flag
	rjmp CONTINUE ;continue once bit 1 is set
	rjmp POLL_1 ;loop until bit 1 is set

CONTINUE:
	ldi r16,0xD8 ;load 'D8' to r16 to enable IOREG
	sts CPU_CCP, r16 ;write IOREG to CPU_CCP

	ldi r16, 0x01 ;load '1' to bit 1
	sts CLK_CTRL,r16 ;store '1' in the CLK_CTRL register to enable the 32 MHz clk

	pop r16
	ret
```

```
;***************************************************
USART_INIT:
	push r16

	ldi r16, 0x08
	sts PORTC_DIRSET, r16

	ldi r16, 0x06
	sts PORTC_DIRCLR, r16

	ldi r16, 0b00110011  ;Asynchronus, Odd Parity Bit, 1 stop bit and 8 data bits
	sts USARTC0_CTRLC, r16

	ldi r16, 0b10101100                    ;lower byte of the Baud Select value

	sts USARTC0_BAUDCTRLA, r16

	ldi r16, 0b10010001  ;high nibble to 1001 for two's comp of -7, top of BSEL to 0001

	sts USARTC0_BAUDCTRLB, r16

	ldi r16, 0b00001000                    ;Transmit enable
	sts USARTC0_CTRLB, r16

	pop r16

	ret

;***************************************************

OUT_CHAR:
	push r16
	push r17

OUT_CHAR_WAIT:
	lds  r16, USARTC0_STATUS
	sbrs r16, 5                            ;check if DATA register is empty
	rjmp OUT_CHAR_WAIT                     ;loop to wait for output enable

	sts USARTC0_DATA, r17                  ;store the character in the data register

	pop r17
	pop r16

	ret
```

Part D:

Pseudocode:

1. Initialize the Clock for 32MHz
2. Initalize the USART for Transmit Enable
3. Initialize a Z pointer for a Table with the Characters of my name
4. Call OUT_STRING subroutine

5. In this subroutine, check and make sure that the value loaded into the register by the Z pointer is not equal to the EOT condition, if it is: don't output anything, return from subroutine, and put the program in endless loop.
6. If it's not the EOT condition, output the current character and increment Z, return from subroutine and enter endless loop so the program doesn't output anything else.

Program Code:

```
; Lab 4 Part D
; Name: Raymond Salzmann
; Section #: 2B04
; TA Name: Keith Fitzgerald
; Description: STRING OUTPUT

.include "ATxmega128A1Udef.inc"

.org 0x0000
	rjmp MAIN

.org 0x2000
	Input_Table: .db 0x10,"RAYMOND", 0x20, "SALZMANN", 0x0A

.cseg

.org 0x200
MAIN:
	call CLOCK_32MHz_SETUP
	call USART_INIT

	ldi ZH, high (Input_Table << 1)
	ldi ZL, low (Input_Table << 1)

	call OUT_STRING

DONE:
	rjmp DONE

;*****************************************************
CLOCK_32MHz_SETUP:
	push r16

SETUP:
	ldi r16, 0x02 ;set bit 1
	sts OSC_CTRL, r16 ;enable OSC_CTRL with the 32 MHz internal Clock

POLL_1:
	lds r16,OSC_STATUS ;load the value of OSC_STATUS to r16
	sbrc r16, 1 ;poll the OSC_STATUS register for stable flag
	rjmp CONTINUE ;continue once bit 1 is set
	rjmp POLL_1 ;loop until bit 1 is set

CONTINUE:
	ldi r16,0xD8 ;load 'D8' to r16 to enable IOREG
	sts CPU_CCP, r16 ;write IOREG to CPU_CCP
```

```
	ldi r16, 0x01 ;load '1' to bit 1
	sts CLK_CTRL,r16 ;store '1' in the CLK_CTRL register to enable the 32 MHz clk

	pop r16
	ret

;*******************************************
USART_INIT:
	push r16

	ldi r16, 0x08
	sts PORTD_DIRSET, r16

	ldi r16, 0x06
	sts PORTD_DIRCLR, r16

	ldi r16, 0b00110011
	sts USARTD0_CTRLC, r16

	ldi r16, 0b10101100                    ;lower byte of the Baud Select value

	sts USARTD0_BAUDCTRLA, r16

	ldi r16, 0b10010001  ;high nibble to 1001 for two's comp of -7, top of BSEL to 0001

	sts USARTD0_BAUDCTRLB, r16

	ldi r16, 0b00001000                    ;Transmit enable
	sts USARTD0_CTRLB, r16

	pop r16

	ret

;*****************************************
OUT_CHAR:
	push r16
	push r17

OUT_CHAR_WAIT:
	lds  r16, USARTD0_STATUS
	sbrs r16, 5                            ;check if DATA reg is empty
	rjmp OUT_CHAR_WAIT                     ;loop to wait

	sts USARTD0_DATA, r17                  ;store the character in the data register

	pop r17
	pop r16

	ret

;*******************************************
OUT_STRING:
	push r16
	push r17

LOOP:
```

```
        elpm r17, Z+          ;load new character
        cpi r17, 0x0A         ;check if its EOT
        breq TEST                     ;if it is skip the OUT_CHAR
        call OUT_CHAR         ;call subroutine
        rjmp LOOP                     ;loop until EOT is met

TEST:
        pop r17
        pop r16

        ret
```

Part E:

Pseudocode:

1. Initialize Clock to 32MHz
2. Initialize the USART to initially only have Transmit Enable
3. Enter infinite loop which will loop calling IN_CHAR and OUT_CHAR
4. IN_CHAR will wait until a new key is pressed to finish the subroutine, loading r17 with the value of the data from the keyboard, toggle receive enable and disable in this subroutine
5. OUT_CHAR will output the character from the IN_CHAR subroutine

Program Code:

```
; Lab 4 Part E
; Name: Raymond Salzmann
; Section #: 2B04
; TA Name: Keith Fitzgerald
; Description: CHARACTER INPUT

.include "ATxmega128A1Udef.inc"

.org 0x0000
        rjmp MAIN

.org 0x200
MAIN:
        call CLOCK_32MHz_SETUP
        call USART_INIT

LOOP:
        call IN_CHAR
        call OUT_CHAR
        rjmp MAIN

;*****************************************************
CLOCK_32MHz_SETUP:
        push r16

SETUP:
        ldi r16, 0x02 ;set bit 1
        sts OSC_CTRL, r16 ;enable OSC_CTRL with the 32 MHz internal Clock
```

```
POLL_1:
	lds r16,OSC_STATUS ;load the value of OSC_STATUS to r16
	sbrc r16, 1 ;poll the OSC_STATUS register for stable flag
	rjmp CONTINUE ;continue once bit 1 is set
	rjmp POLL_1 ;loop until bit 1 is set

CONTINUE:
	ldi r16,0xD8 ;load 'D8' to r16 to enable IOREG
	sts CPU_CCP, r16 ;write IOREG to CPU_CCP

	ldi r16, 0x01 ;load '1' to bit 1
	sts CLK_CTRL,r16 ;store '1' in the CLK_CTRL register to enable the 32 MHz clk

	pop r16
	ret

;************************************************************
USART_INIT:
	push r16

	ldi r16, 0x08
	sts PORTD_DIRSET, r16

	ldi r16, 0x06
	sts PORTD_DIRCLR, r16

	ldi r16, 0b00110011
	sts USARTD0_CTRLC, r16

	ldi r16, 0b10101100                    ;lower byte of the Baud Select value

	sts USARTD0_BAUDCTRLA, r16

	ldi r16, 0b10010001  ;high nibble to 1001 for two's comp of -7, top of BSEL to 0001

	sts USARTD0_BAUDCTRLB, r16

	ldi r16, 0b00001000                    ;Transmit enable
	sts USARTD0_CTRLB, r16

	pop r16

	ret

;************************************************************

OUT_CHAR:
	push r16
	push r17

OUT_CHAR_WAIT:
	lds  r16, USARTD0_STATUS
	sbrs r16, 5                        ;check if DATA is ready to recieve new empty
	rjmp OUT_CHAR_WAIT                 ;loop to wait for DATA to be ready

	sts USARTD0_DATA, r17      ;write the character into the data register
```

```
        pop r17
        pop r16

        ret

;************************************************************
IN_CHAR:
        push r16

IN_CHAR_WAIT:
        lds  r16, USARTD0_STATUS
        sbrs r16, 5                ;check if DATA is empty
        rjmp IN_CHAR_WAIT                       ;loop to wait for DATA to be ready

        ldi r16, 0b00011000                     ;Receive enable
        sts USARTD0_CTRLB, r16

TEST:
        lds r17, USARTD0_DATA
        cpi r17, 0x00
        breq TEST                       ;check if DATA is ready to recieve new empty

DONE:
        ldi r16, 0b00001000                     ;Receive disable
        sts USARTD0_CTRLB, r16

        pop r16

        ret
```

Part F:

Pseudocode:

1. Setup tables which will contain every string of the menu
2. Initialize the Clock to 32Mhz
3. Initialize the USART
4. Reset the Z pointer to point to the table with the menu
5. Output the Menu for the first time
6. Infinite loop the call OUT_STRING and check to see if done was pressed
7. Check if the value is EOT, if it is have the program stop outputting, if its not, use OUT_CHAR and repeat
8. Have a subroutine called CHECK that is called in the OUT_STRING subroutine which checks the output of the IN_CHAR subroutine, and loads the Z pointer to the different tables, or does nothing if anything else besides what's on the menu is pressed

Program Code:

```
; Lab 4 Part F
; Name: Raymond Salzmann
; Section #: 2B04
```

```
; TA Name: Keith Fitzgerald
; Description: SERIAL MENU

.include "ATxmega128A1Udef.inc"

.equ CR = 0x0D
.equ LF = 0x0A

.org 0x0000
	rjmp MAIN

.org 0x2000
;set up a table for every possible menu/answer
	Input_Table: .db 0x10,"RAYMOND SALZMANN'S FAVORITE:",CR,LF,"1. Food",CR,LF,"2. UF
Course",CR,LF,"3. Hobby",CR,LF,"4. Quote",CR,LF,"5. Movie",CR,LF,"6. Re-display
menu",CR,LF,"D: Done",CR,LF, 0xFF
	HOBBY: .db "RAYMOND SALZMANN'S FAVORITE HOBBY IS PLAYING VIDEO GAMES",CR,LF,0xFF
	FOOD: .db "RAYMOND SALZMANN'S FAVORITE FOOD IS ONE DANK CHICKEN TACO",CR,LF,0xFF
	COURSE: .db "RAYMOND SALZMANN'S FAVORITE UF COURSE IS SOLID STATE ELECTRONIC
DEVICES",CR,LF,0xFF
	QUOTE: .db 0x10, "RAYMOND SALZMANN'S FAVORITE QUOTE IS: THE ONLY THING WE HAVE TO
FEAR IS FEAR ITSELF",CR,LF,0xFF
	MOVIE: .db "RAYMOND SALZMANN'S FAVORITE MOVIE IS THE SHINING",CR,LF,0xFF
	DONE_Table: .db "Done!",CR,LF,0xFF

.cseg

.org 0x200
MAIN:
	call CLOCK_32MHz_SETUP
	call USART_INIT
	call RESET
	call MENU_RESET

LOOP:
	call OUT_STRING

	cpi r21, 0x01 ;if r21 is 1, from the DONE part of the menu, then end the program
	breq DONE
	rjmp LOOP

DONE:
	rjmp DONE

;*****************************************************
CLOCK_32MHz_SETUP:
	push r16

SETUP:
	ldi r16, 0x02 ;set bit 1
	sts OSC_CTRL, r16 ;enable OSC_CTRL with the 32 MHz internal Clock

POLL_1:
	lds r16,OSC_STATUS ;load the value of OSC_STATUS to r16
	sbrc r16, 1 ;poll the OSC_STATUS register for stable flag
	rjmp CONTINUE ;continue once bit 1 is set
	rjmp POLL_1 ;loop until bit 1 is set
```

```
CONTINUE:
	ldi r16,0xD8 ;load 'D8' to r16 to enable IOREG
	sts CPU_CCP, r16 ;write IOREG to CPU_CCP

	ldi r16, 0x01 ;load '1' to bit 1
	sts CLK_CTRL,r16 ;store '1' in the CLK_CTRL register to enable the 32 MHz clk

	pop r16
	ret

;*********************************************
USART_INIT:
	push r16

	ldi r16, 0x08
	sts PORTD_DIRSET, r16

	ldi r16, 0x06
	sts PORTD_DIRCLR, r16

	ldi r16, 0b00110011
	sts USARTD0_CTRLC, r16

	ldi r16, 0b10101100		;lower byte of the Baud Select value

	sts USARTD0_BAUDCTRLA, r16

	ldi r16, 0b10010001  ;high nibble to 1001 for two's comp of -7, top of BSEL to 0001

	sts USARTD0_BAUDCTRLB, r16

	ldi r16, 0b00001000			;Transmit enable
	sts USARTD0_CTRLB, r16

	pop r16

	ret

; ************************************************
OUT_CHAR:
	push r16
	push r17

OUT_CHAR_WAIT:
	lds  r16, USARTD0_STATUS
	sbrs r16, 5				;check if DATA reg is empty
	rjmp OUT_CHAR_WAIT			;loop to wait for output enable

	sts USARTD0_DATA, r17			;store the character in the data register

	pop r17
	pop r16

	ret

; ************************************************

OUT_STRING:
```

```
        push r16
        push r17

        call CHECK

        cpi r20, 0x01               ;check for menu reset
        breq OUT_STRING_END

OUT_STRING_REPEAT:
        elpm r17, Z+                ;load r17 with the table data

        cpi  r17, 0xFF                      ;check for null value
        breq OUT_STRING_END

        call OUT_CHAR               ;call OUT_CHAR to output the character
        rjmp OUT_STRING_REPEAT

OUT_STRING_END:

        cpi r21, 0x01
        BREQ DONE

        call MENU_RESET

        ldi r16,0x00
        cpse r20, r16               ;check if the menu was called and reset to zero
        ldi     r20, 0x00

        pop r17
        pop r16

        ret

; ************************************
RESET:
        ldi ZH, High(Input_Table << 1);load the high byte of the Table address into ZH
        ldi ZL, Low(Input_Table << 1);load the low byte of the Table address into ZL

        ret
; ************************************
CHECK:
        call IN_CHAR

        cpi r17, '1'                ;check if 1 is input
        breq CHAR_ONE

        cpi r17, '2'                ;check if 2 is input
        breq CHAR_TWO

        cpi r17, '3'                ;check if 3 is input
        breq CHAR_THREE

        cpi r17, '4'                ;check if 4 is input
        breq CHAR_FOUR

        cpi r17, '5'                ;check if 5 is input
        breq CHAR_FIVE
```

```
        cpi r17, '6'                ;check if 6 is input
        breq CHAR_SIX

        cpi r17, 'D'                ;check if D is input
        breq CHAR_DONE

        cpi r17, 'd'                ;check if d is input
        breq CHAR_DONE

        jmp LOOP

CHAR_ONE:
        ldi ZH, High(FOOD << 1);load the high byte of the FOOD Table address into ZH
        ldi ZL, Low(FOOD << 1)      ;load the low byte of the FOOD Table address into ZL
        rjmp CHAR_CHECK_DONE

CHAR_TWO:
        ldi ZH, High(COURSE << 1) ;load the high byte of the COURSE Table address into ZH
        ldi ZL, Low(COURSE << 1) ;load the low byte of the COURSE Table address into ZL
        rjmp CHAR_CHECK_DONE

CHAR_THREE:
        ldi ZH, High(HOBBY << 1) ;load the high byte of the HOBBY Table address into ZH
        ldi ZL, Low(HOBBY << 1) ;load the low byte of the HOBBY Table address into ZL
        rjmp CHAR_CHECK_DONE

CHAR_FOUR:
        ldi ZH, High(QUOTE << 1) ;load the high byte of the QUOTE Table address into ZH
        ldi ZL, Low(QUOTE << 1) ;load the low byte of the QUOTE Table address into ZL
        rjmp CHAR_CHECK_DONE

CHAR_FIVE:
        ldi ZH, High(MOVIE << 1) ;load the high byte of the MOVIE Table address into ZH
        ldi ZL, Low(MOVIE << 1)     ;load the low byte of the MOVIE Table address into ZL
        rjmp CHAR_CHECK_DONE

CHAR_SIX:
        ldi r20, 0x01                           ;load r20 with one to signal menu reload
        rjmp CHAR_CHECK_DONE

CHAR_DONE:
        ldi ZH, High(DONE_Table << 1) ;load the high byte of the DONE_Table into ZH
        ldi ZL, Low(DONE_Table << 1) ;load the low byte of the DONE_Table address into ZL
        ldi r21, 0x01
        rjmp CHAR_CHECK_DONE

CHAR_CHECK_DONE:
        ret

; ****************************************
IN_CHAR:
        push r16

IN_CHAR_WAIT:
        lds  r16, USARTD0_STATUS
        sbrs r16, 5                             ;check if DATA reg is empty
        rjmp IN_CHAR_WAIT                       ;loop to wait for output enable
```

```
        ldi r16, 0b00011000                     ;Receive enable
        sts USARTD0_CTRLB, r16

IN_CHAR_CHECK:

        lds r17, USARTD0_DATA                   ;value to be written
        cpi r17, 0x00
        breq IN_CHAR_CHECK                      ;check for input, loop if none

        ldi r16, 0b00001000                     ;Receive disable
        sts USARTD0_CTRLB, r16

        pop r16

        ret

;*****************************************
MENU_RESET:
        push r16
        push r17

        call RESET

MENU_RESET_OUT_STRING_REPEAT:
        elpm r17, Z+                ;load r17 with value pointed to by Z

        cpi  r17, 0xFF                      ;check for EOT Condition
        breq MENU_RESET_OUT_STRING_END

        call OUT_CHAR               ;call OUT_CHAR to output the character
        rjmp MENU_RESET_OUT_STRING_REPEAT

MENU_RESET_OUT_STRING_END:
        pop r17
        pop r16

        ret
```

Part G:

Pseudocode:

1. Initialize clock to 32 MHz
2. Initialize LED PORTD
3. Initialize the TC for blinking
4. Initialize USART
5. Set up the ISR for transmitting/receiving using the IN_CHAR and OUT_CHAR subroutine code from past parts
6. Set up an ISR that relies on the overflow of the TC for blinking the LED

Program Code:

```
; Lab 4 Part G
; Name: Raymond Salzmann
; Section #: 2B04
; TA Name: Keith Fitzgerald
; Description: INTERRUPT DRIVEN RECEIVING

.include "ATxmega128A1Udef.inc"

.org 0x0000
	rjmp MAIN

.org USARTD0_RXC_vect
	jmp PART_E_ISR

.org TCE0_OVF_vect
	jmp LOGIC

.cseg

.org 0x200
MAIN:
	call CLOCK_32MHz_SETUP
	call USART_INIT
	call INITIALIZE
	call TIMER_COUNTER

LOOP:
	rjmp LOOP

;*********************************************************
CLOCK_32MHz_SETUP:
	push r16

SETUP:
	ldi r16, 0x02 ;set bit 1
	sts OSC_CTRL, r16 ;enable OSC_CTRL with the 32 MHz internal Clock

POLL_1:
	lds r16,OSC_STATUS ;load the value of OSC_STATUS to r16
	sbrc r16, 1 ;poll the OSC_STATUS register for stable flag
	rjmp CONTINUE ;continue once bit 1 is set
	rjmp POLL_1 ;loop until bit 1 is set

CONTINUE:
	ldi r16,0xD8 ;load 'D8' to r16 to enable IOREG
	sts CPU_CCP, r16 ;write IOREG to CPU_CCP

	ldi r16, 0x01 ;load '1' to bit 1
	sts CLK_CTRL,r16 ;store '1' in the CLK_CTRL register to enable the 32 MHz clk

	pop r16
	ret

;*********************************************************
USART_INIT:
	push r16
```

```
        ldi r16, 0x08
        sts PORTD_DIRSET, r16

        ldi r16, 0x06
        sts PORTD_DIRCLR, r16

        ldi r16, 0b00110011
        sts USARTD0_CTRLC, r16

        ldi r16, 0b10101100                     ;lower byte of the Baud Select value

        sts USARTD0_BAUDCTRLA, r16

        ldi r16, 0b10010001 ;high nibble to 1001 for two's comp of -7, top of BSEL to 0001

        sts USARTD0_BAUDCTRLB, r16

        ldi r16, 0b00011000                     ;Transmit & Receive enable
        sts USARTD0_CTRLB, r16

        ldi r17, 0b00010000
        sts USARTD0_CTRLA, r16

        ldi r16, 0x01                           ;load PMIC_CTRL with a low level
        sts PMIC_CTRL, r16

        sei

        pop r16
        ret

;*************************************************************
PART_E_ISR:
        push r16
        push r17

        lds r17, USARTD0_DATA

        sts USARTD0_DATA, r17       ;write the character into the data register

        ldi r16, 0x80
        sts USARTD0_STATUS, r16

        pop r17
        pop r16

        reti

;*************************************************************
INITIALIZE:
        push r16

        ldi r16, 0b00100000                     ;enable PORTD as outputs
        sts PORTD_DIRSET, r16

        ldi r18, 0x00
```

```
	pop r16
	ret

;************************************************************
TIMER_COUNTER:
	push r16
	push r17

	ldi r16, 0x00 ;load 00 into the low byte of the PER register
	sts TCE0_PER, r16

	ldi r16, 0x45 ;load 02 into the high byte of the PER register
	sts TCE0_PER+1, r16

	ldi r16, 0x00 ;initialize the counter at zero
	sts TCE0_CNT, r16

	ldi r16, 0x00 ;initialize the counter at zero
	sts TCE0_CNT+1, r16

	ldi r16, 0x01 ;enable the overflow interrupt
	sts TCE0_INTCTRLA, r16

	ldi r16, 0x07 ;load 7 into r16 for the slowest TC prescaler
	sts TCE0_CTRLA, r16

	pop r17
	pop r16
	ret

;************************************************************
LOGIC:
	push r16
	push r17
	cpi r18, 0x00
	breq ON
	rjmp OFF

ON:
	ldi r16, 0x00
	sts PORTD_OUT, r16
	ldi r18, 0x01
	rjmp CLEAR_FLAG

OFF:
	ldi r16, 0xFF
	sts PORTD_OUT, r16
	ldi  r18, 0x00
	rjmp CLEAR_FLAG

CLEAR_FLAG:

	ldi r16, 0x01              ;write a 1 to int0 to reset it
	sts TCE0_INTFLAGS, r16

	pop r17
	pop r16
	reti
```

Appendix

Full Character transmission (Falling Edge of wider pulse to Rising Edge of next wider pulse):

Single Bit transmission: